

# Overview

🕒 2 minute read

---

---

Distributed tracing enables users to track a request through mesh that is distributed across multiple services. This allows a deeper understanding about request latency, serialization and parallelism via visualization.

Istio leverages Envoy's distributed tracing feature to provide tracing integration out of the box. Specifically, Istio provides options to install various tracing backend and configure proxies to send trace spans to

them automatically. See [Zipkin](#), [Jaeger](#) and [Lightstep](#) task docs about how Istio works with those tracing systems.

## **Trace context propagation**

Although Istio proxies are able to automatically send spans, they need some hints to tie together the entire trace. Applications need to propagate the appropriate HTTP headers so that when the proxies send span information, the spans can be correlated correctly into a single trace.

To do this, an application needs to collect and propagate the following headers from

the incoming request to any outgoing requests:

- `x-request-id`
- `x-b3-traceid`
- `x-b3-spanid`
- `x-b3-parentspanid`
- `x-b3-sampled`
- `x-b3-flags`
- `x-ot-span-context`

Additionally, tracing integrations based on OpenCensus (e.g. Stackdriver) propagate the following headers:

- `x-cloud-trace-context`
- `traceparent`
- `grpc-trace-bin`

If you look at the sample Python [productpage](#)

service, for example, you see that the application extracts the required headers from an HTTP request using OpenTracing libraries:

```

def getForwardHeaders(request):
    headers = {}

    # x-b3-*** headers can be populated using the o
    pentracing span
    span = get_current_span()
    carrier = {}
    tracer.inject(
        span_context=span.context,
        format=Format.HTTP_HEADERS,
        carrier=carrier)

    headers.update(carrier)

    # ...

    incoming_headers = ['x-request-id', 'x-datadog-
    trace-id', 'x-datadog-parent-id', 'x-datadog-sample
    d']

    # ...

    for ihdr in incoming_headers:
        val = request.headers.get(ihdr)
        if val is not None:
            headers[ihdr] = val

    return headers

```

The reviews application (Java) does

something similar using requestHeaders:

```
@GET
@Path("/reviews/{productId}")
public Response bookReviewsById(@PathParam("product
Id") int productId, @Context HttpHeaders requestHea
ders) {

    // ...

    if (ratings_enabled) {
        JsonObject ratingsResponse = getRatings(Integer
.toString(productId), requestHeaders);
```

When you make downstream calls in your applications, make sure to include these headers.